VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

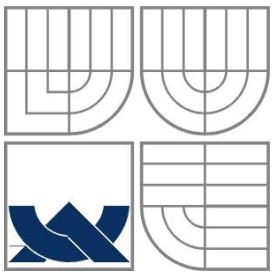
INFORMAČNÍ SYSTÉM PRO PODPORU VYHODNOCOVÁNÍ SOUTĚŽE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

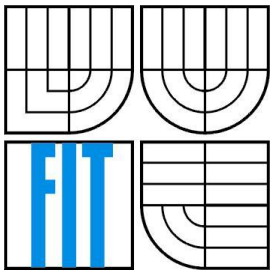
AUTOR PRÁCE
AUTHOR

ČENĚK KOVÁŘ

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO PODPORU VYHODNOCOVÁNÍ SOUTĚŽE

INFORMATION SYSTEM FOR SUPPORT OF COMPETITION EVALUATION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

ČENĚK KOVÁŘ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ STRYKA

BRNO 2008

Abstrakt

Cílem této práce je navrhnout a v prostředí .NET vytvořit informační systém pro podporu Svojsíkova závodu. Tento systém se skládá ze dvou částí – tlustého a tenkého klienta. Jeho hlavním účelem je poskytnout informace o průběhu a výsledcích jednotlivých kol.

Klíčová slova

Informační systém, ASP.NET, .NET, C#, tlustý klient, tenký klient, databáze, XML, XHTML, JavaScript, web

Abstract

The goals of this bachelor‘s thesis are design and implementation of information system in .NET framework. This system contains two parts – web interface and desktop application. The goal of this system is to provide information about individual rounds results.

Keywords

Information system, ASP.NET, .NET, C#, web interface, desktop application, database, XML, XHTML, JavaScript, web

Citace

Kovář Čeněk: Informační systém pro podporu vyhodnocování soutěže. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Informační systém pro podporu vyhodnocování soutěže

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Lukáše Stryky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Čeněk Kovář
14.5.2008

Poděkování

Rád bych poděkoval všem lidem, kteří mi pomohli při řešení této bakalářské práce a také těm, kteří mě během jejího psaní jakkoliv podpořili. Chtěl bych také poděkovat svému vedoucímu Ing. Lukáši Strykovi.

Obsah

Úvod

Cílem této bakalářské práce bylo navrhnout a vytvořit informační systém pro organizaci Junák – svaz skautů a skautek ČR, který by umožňoval spravovat výsledky a usnadňoval organizaci jednotlivých kol Svojsíkova závodu. Měl také nabídnout přehledné webové rozhraní, kde by uživatel našel všechny potřebné informace o jednotlivých kolech, datum a místo konání atd.

Výsledná aplikace se skládá ze dvou vzájemně provázaných částí – tenkého a tlustého klienta, kdy tenký klient běží na webovém serveru a uživatel k němu získává přístup pomocí internetového prohlížeče a tlustý klient je klasická windows aplikace, která běží přímo na počítači uživatele.

Celkově je práce rozdělena do několika kapitol, kdy se každá zabývá specifickou problematikou a tvoří jakýsi tématický okruh.

První kapitola si klade za cíl vysvětlit co je Svojsíkův závod, stručně nastínit jeho pravidla a popsat způsob, kterým je hodnocen.

Druhou částí je neformální specifikace systému. Ta souhrnně popisuje funkcionalitu celé aplikace, součinnost obou klientů a další věci, které nejsou ryze technického směru.

Cílem další kapitoly je uvést technologie, jež byly použity při implementaci řešení. Konkrétně .NET Framework, ASP.NET a jazyk C#. Je to pouze úvod do celé problematiky.

Ve čtvrté části se zaměřím na podrobnější rozebrání tenkého klienta, jeho návrh, strukturu databáze i to, jakým způsobem je řešeno předávání parametrů pomocí jinak bezestavového protokolu HTTP, celkové zabezpečení aplikace, role uživatelů atd.

Pátou kapitolu jsem naopak věnoval tlustému klientovi, především zpracování dat, jejich vyhodnocení a následný export do uživatelsky přívětivé podoby.

1 Svojsíkův závod

Svojsíkův závod[1] (dále jen SZ) je závodem skautských hlídek – skupinek složených z mladých lidí, členů organizace Junák – svaz skautů a skautek ČR. SZ je hierarchicky uspořádán a postup do vyššího kola si hlídka musí zajistit postupem z kola nižšího. Podle počtu hlídek v závodě se určí i počet postupujících do dalšího kola. Nejnižším kolem je základní, odtud hlídka může postoupit do krajského a poté i celostátního kola, které se koná jednou za dva roky.

Závod se sestává z několika modulů, které jsou jakýmsi obalem pro bodovaná stanoviště. Modul „Závod“ je povinný ve všech kolech. Další moduly „Brány“, „Výpravy“ a „Táboření“ jsou nepovinné, vždy však musí být v závodě alespoň dva. Jejich minimální počet je ale dán typem závodu. V základním kole je to modul „Závod“ a jeden ze zbylých modulů, v krajském kole jsou již povinné dva z volitelných modulů a celostátní kolo musí obsahovat moduly všechny. Každý modul má danou procentuální váhu, přičemž za modul „Závod“ musí jít získat vždy alespoň 50% bodů.

V každém závodě může hlídka v součtu získat maximálně 1000 bodů na jednotlivých stanovištích. Hlídka s nejvyšším počtem bodů se stává vítězem. Podle nových pravidel už není tolik zohledňován čas na trati. Pokud ale pořadatel chce, může hlídky, které jsou výrazně rychlejší než stanovený „ideální čas závodu“ odměnit bonusovými body – toto lze realizovat např. uměle vytvořeným stanovištěm.

Po skončení závodu se provádí přepočet bodů, pro který se používá následujícího vzorce:

$$\frac{\text{Získané body za disciplíny v jednom modulu}}{\text{Max. možný zisk bodu v daném modulu}} \times \text{váha modulu} \times 1000$$

Př.: Hlídka v modulu „Závod“ získá za body ze stavišť v tomto modulu 350 bodů. Maximální možný bodový zisk byl 450 bodů. Modul „Závod“ byl zatížen 60% (0,6). Družina tedy za tento modul „Závod“ získává do celkového hodnocení 466.7 bodů

$$\frac{350}{450} \times 0{,}6 \times 1000 = 466.7$$

[1] Úplné znění posledních pravidel Svojsíkova závodu je v příloze.

2 Neformální specifikace

V této kapitole bude popsána neformální specifikace celého systému, která vznikla po konzultacích se zástupci organizace Junák. Budou uvedeny požadavky na systém a funkce, které by měl podporovat.

Úkolem bylo navrhnout a vytvořit informační systém, rozdělený na dva moduly, poskytující informace o průběhu jednotlivých kol Svojsíkova závodu, od nejnižších až po celostátní. Neměl být velkým portálem, na kterém by se shromažďovalo množství informací o jednotlivých kolech, spíše jakýsi rozcestník, kde by byly stručné informace o jednotlivých závodech.

Již bylo zmíněno, systém měl být rozdělen do dvou částí - klientů. Z pohledu klasického uživatele bude určitě zajímavější částí tenký klient ve formě webového klienta. Požadavkem bylo, aby u jednotlivých kol evidovala základní informace potřebné pro identifikaci závodu, jako datum a místo konání každého klání, případně i kraj či oblast, do které závod spadá, pokud se nejedná o státní kolo. U každého závodu měla být možnost prohlížet jednoduchou galerii fotografií, napsat příspěvek do knihy návštěv nebo před vlastním konáním sledovat, které hlídky se k závodu již přihlásily. Dojde li k nějaké nečekané změně ohledně závodu, můžou být uživatelé o této změně informováni pomocí aktualit. V neposlední řadě, jak už bylo zmíněno, byla požadována funkce prohlížet výsledky závodů. To vše na jednom místě. Mělo se tedy jednat o jakýsi doplněk k webovým stránkám jednotlivých oddílů, jež konkrétní kola SZ pořádají. Pro usnadnění organizace měla být implementována i část, kde by si administrátor do systému vkládal hlídky, které již doručily předběžnou přihlášku, umožňovala jednoduchou správu rolí a uživatelů, editaci údajů závodů nebo přidávání fotografií do galerie.

Naopak tlustý klient měl být určen výhradně pro pořadatele závodu, kterým by umožňoval přímo na závodišti zadat hodnocení hlídkám i bez nutnosti přístupu k internetu a spojení s tenkým klientem. Přitom také kontrolovat tyto hodnoty, aby například nedocházelo k překlepům v řádu přidělených bodů apod. Na konci by provedl součet bodů a export dat pro jejich další prezentaci. Dalším požadavkem byla implementace jednoduché správa závodníků sloužící především jako jakési vodítko pro pořadatele s možností vyhledávání osob.

3 Použité technologie

Celé řešení je implementováno v prostředí .NET framework [6] firmy Microsoft. Oba klienti byli implementováni v programovacím jazyku C#, neboť .NET podporuje CLR. Kromě těchto technologií byly zejména v rámci tenkého klienta aplikovány i značkovací jazyky HTML, XML, skriptovací jazyk JavaScript a XLST spolu CSS pro formátování výsledného vzhledu.

3.1 ASP.NET

ASP.NET není programovacím jazykem. Je součástí .NET framework a lze v něm tvořit aplikace určené pro webové prostředí. Ačkoli je názvem podobný ASP, jde o technologii poměrně odlišnou. Zatímco ASP bylo klasickým příkladem skriptovacího stroje, je ASP.NET založen (stejně jako celý .NET) na CLR – *Common Language Runtime*. Z toho plyne i veliká výhoda, že programovacím jazykem v ASP.NET může být jakýkoliv jazyk podporující CLR. Mezi nejpoužívanější rozhodně patří Visual Basic .NET, C# ale i starší jazyky typu pascal. Dalo by se říci, že ASP.NET je kompletním objektovým modelem pro webové stránky, který řídí veškeré procesy na základě událostí. Tímto se začalo programování internetových aplikací [1, 2] blížit klasickému programování desktop aplikací [3]. Samozřejmě určité rozdíly plynou z použitého bezestavového protokolu HTTP. Uživatel může aplikaci kdykoliv ukončit, procházet historií stránek zpět či vpřed apod. To vše aniž by měla aplikace možnost jakkoliv zareagovat. Pro událostmi řízené prostředí je však třeba, aby se někde uchovávaly informace o stavu a hodnotě jednotlivých objektů na stránce. K tomu je v ASP.NET využíváno tří zásadních prostředků – SessionState, PostBack a ViewState. PostBack znamená, že při každé interakci s uživatelem – např. kliknutí na tlačítko – se formulář odešle metodou POST na server a stránka zpracovává jakoby sama sebe. Jádro ASP.NET se na serveru samo stará o zjištění stavu jednotlivých prvků a vyvolání příslušných ovladačů událostí. Druhým důležitým podpojmem je ViewState, což je maximálně zhuštěná informace o stavu stránky. Tento prvek je obsažen v každé generované stránce, konkrétně ve skrytém inputu s názvem *__VIEWSTATE*. ViewState se plní při každém PostBacku. Jeho výhodou je použití prostého textu a nevyžaduje tak žádné dodatečné prostředky potřebné pro běh, nevýhodou jsou pak další přenášená data. SessionState dokáže udržet informace při přechodu mezi stránkami. Podrobnější popis session bude popsán v kapitole věnující se tenkému klientovi, který tyto prostředky využívá.

Jak již bylo zmíněno, je ASP.NET plně objektové, takže při tvorbě webových stránek lze pouze umisťovat jednotlivé instance objektů a každý takový webový ovládací prvek produkuje validní HTML kód. Programátor tak nemusí prakticky znát syntaxi HTML.

Díky vzrůstající popularitě ASP.NET lze na internetu nalézt spoustu tutoriálů a diskusí věnujících se tomuto tématu [5, 6]

Tenký klient

Obsahem této kapitoly bude podrobnější popis webové části informačního systému – tzv. *tenkého klienta*. Při návrhu vzhledu a struktury bylo dbáno na celkovou jednoduchost. Důraz byl také kladen na použití kaskádových stylů a jejich oddělení od zdrojového kódu. Pro změnu vzhledu tedy postačí upravit tento stylový předpis a do struktury stránky nemusí být zasahováno. Samozřejmě záleží na rozsahu těchto změn. Současná verze je zachycena na obr. 1.

Obrázek 1 - úvodní obrazovka aplikace

Vzhled není ještě ve finální fázi, protože se klient bude teprve testovat a případné změny a úpravy se tak promítnou až do konečné verze.

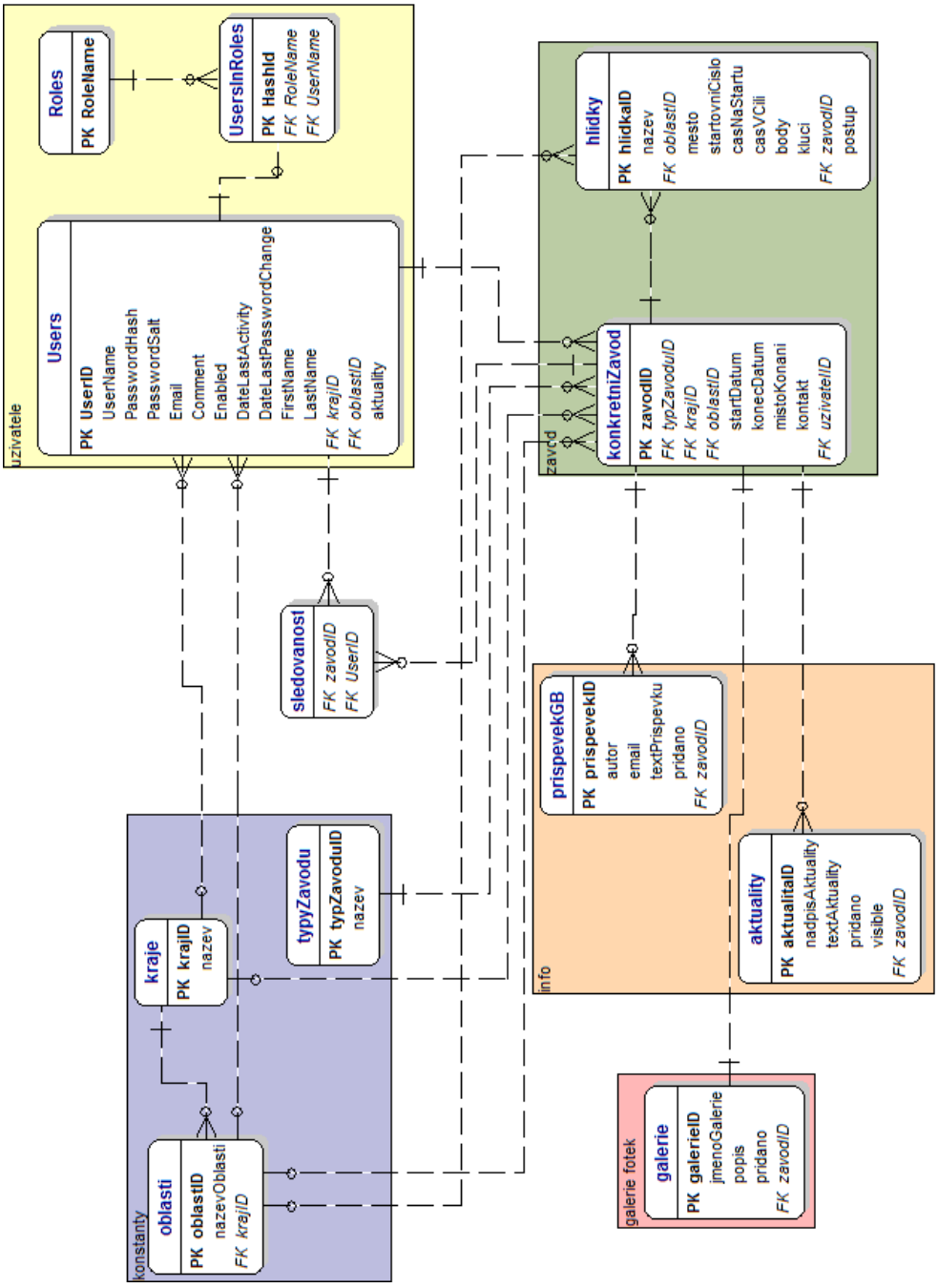
3.2 Návrh databáze

Jako databázový stroj byl použit Microsoft SQL Server 2008, běžící na hostingu, kde je aplikace umístěna. Jako primární klíče byly v databázi použity uměle vytvořené atributy – ID, které jsou typu

INTEGER. Jedná se o celočíselný typ, se kterým databázový stroj vykazuje nejlepší výsledky při indexaci. Snažil jsem se co nejvíce vyhnout redundanci dat a to použitím cizích klíčů. Ty se odkazují na data v jiných tabulkách a pak, ve výsledné aplikaci, lze přes tyto cizí klíče propojit data z obou tabulek – jak odkazované tak i té, která na ni odkazuje.

3.2.1 ER diagram

Pro modelování jednotlivých entit v databázi a vztahů mezi nimi bylo využito ER diagramu – *Entity-relationship diagram*. Jedná se o relační schéma vyjadřující sémantiku vztahů. Tím bylo dosaženo grafického znázornění a náhledu na celou databázi. Celý ER diagram je na obrázku 2.



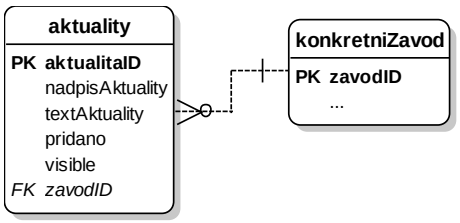
Obrázek 2 – celkový pohled na ER-diagram

Analýza použitých tabulek

Následný seznam slouží k upřesnění a krátkému popisu jednotlivých entit databáze a jejich atributů.

aktuality

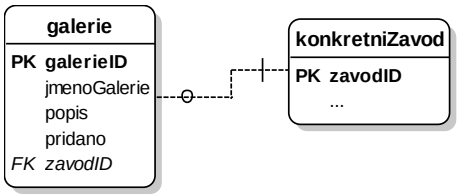
Tato entita sdružuje všechny informace o aktualitách, které chce administrátor rozšířit mezi uživatele. Každá taková novinka se váže ke konkrétnímu závodu, ke kterému je vázána pomocí cizího klíče. Spojení obou entit je jasné z obrázku 3.



Obrázek 3 – zobrazení entity aktuality + její vztah s okolím

galerie

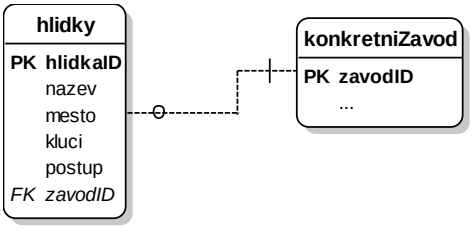
V původním návrhu aplikace byla i pokročilejší práce s fotogalerií, ale nakonec bylo od její realizace upuštěno. Tato entita zde ale zůstala a udržuje informace o jednotlivých galeriích, viz obr. 4. Jde čistě o informační záležitosti, které nemají žádnou souvislost s fyzickým umístěním obsahu galerie na serveru.



Obrázek 4 – entita galerie a vztah k okolí

hlidky

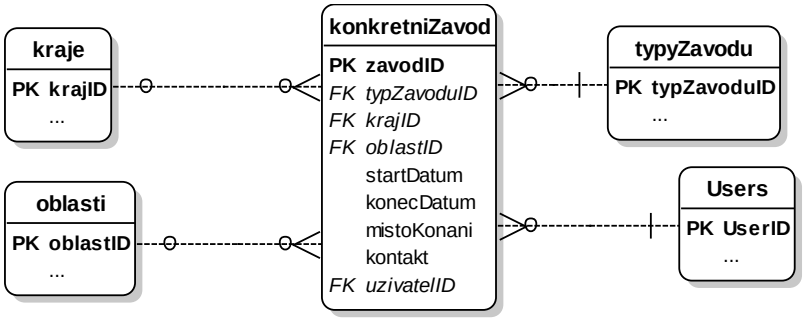
Tabulka *hlidky* byla původně určena pouze k uchování informací o hlídce, její poloze a kategorii, do které spadá. Postupně se rozšířila i o hodnocení hlídky v závodě, její bodový zisk a vše s těmito úkony související. Od toho ale bylo ve finální fázi (viz obr. 5) opět upuštěno a výsledky jsou uchovány zcela mimo databázi.



Obrázek 5 – vztah mezi entitami hlidky a konkretniZavod

konkretniZavod

V této tabulce, která je znázorněna na obrázku č. 6, jsou uchovány údaje pro jednotlivá kola soutěže. Opět je použito cizích klíčů pro svázání tabulky s rodičovskou tabulkou (kraje, oblasti, typyZavodu, uzivatelID). Zde jen doplním, že atribut uzivatelID identifikuje uživatele, který závod založil a má tedy právo jej upravovat.



Obrázek 6 – vztah konkretniZavod a okolních entit

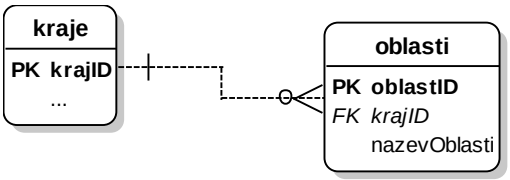
kraje

jednoduchá tabulka (viz obr. 7) udržující pouze seznam krajů České republiky. To usnadňuje řazení jednotlivých kol SZ do hierarchické struktury, která je popsána výše.

Obrázek 7 – entita kraje

oblasti

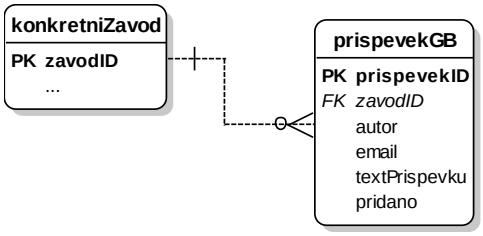
Podobně uspořádaná tabulka jako kraje. Rozdílem je přítomnost cizího klíče, indikujícího spojitost s krajem. Opět pro usnadnění přiřazení závodu do geografické oblasti. Spojení patrno z obr. 8.



Obrázek 8 – vztah entit oblasti a kraje

prispevekGB

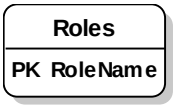
Entita prispevekGB (obr. 9) sdružuje všechny příspěvky, které uživatelé zadali do systému. Přes identifikátor zavodID se lze dostat pouze k požadovaným příspěvkům vážícím se na požadovaný závod. Opět, zbytek názvů atributů je dostatečně popisný.



Obrázek 9 – vztah mezi entitami prispevekGB a konkretniZavod

Roles

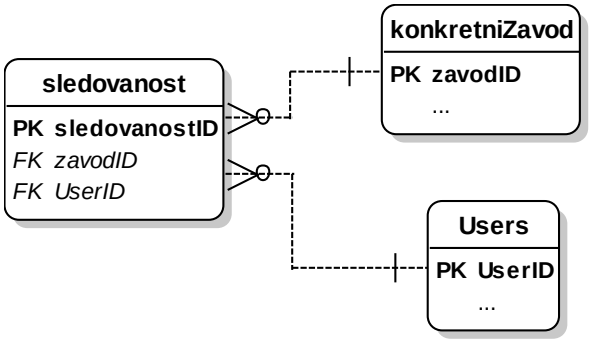
Tato tabulka byla součástí systému přihlašování, které je v aplikaci použita. Není zde použito celočíselného identifikátoru, ale pouze názvu role (viz obr. 10). K tomuto řešení jsem přistupoval „tak jak je“ a snažil jsem se do něho zasahovat co možná nejméně.



Obrázek 10 – entita Roles

Sledovanost

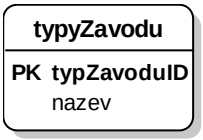
Jak již bylo zmíněno, uživatel se může přihlásit ke sledování závodu. A právě k uchování těchto informací, kdo sleduje který závod, slouží tato tabulka znázorněná na obrázku 11.



Obrázek 11 – vztah entit sledovanost, konkretniZavod a Users

typyZavodu

Při popisu hierarchické struktury byla řeč o tom, že SZ je diferencován do určité struktury na sebe navazujících úrovní. Entita typyZavodu je informační tabulkou, ve které jsou uloženy právě tyto typy závodů, viz obr. 12.

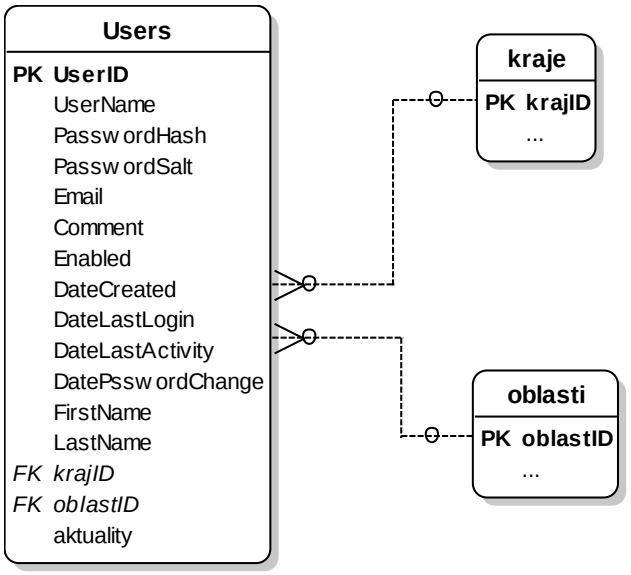


Obrázek 12 – entita typyZavodu

Users

Tabulka Users (obr. 13) je další ze součástí přihlašovacího systému aplikace, který zajišťuje též správu rolí. Opět jsem se snažil do již existující struktury zasahovat co nejméně a přidal jen ty informace, které byly skutečně nezbytné pro běh systému. Konkrétně se jedná o cizí klíče

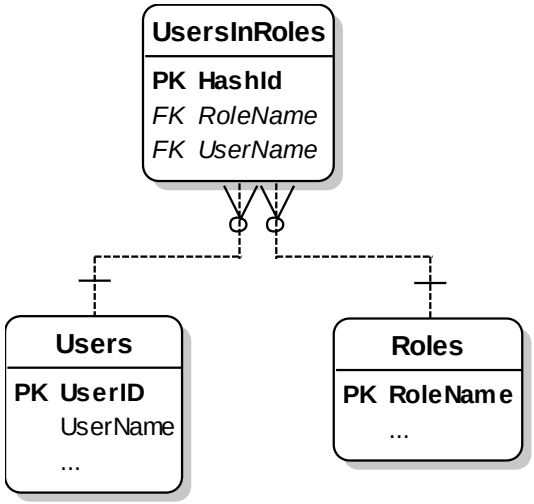
krajID,oblastID a informační atributy FirstName,LastName a aktuality, což je BOOLovský příznak, zda dal uživatel souhlas k odebírání novinek o jednotlivých závodech. Vím, že v tabulce jsou uchovávány i informace, které nejsou nijak využity – jmenujme např. datum vytvoření účtu, poslední přihlášení apod., ale jak jsem již zmínil, snažil jsem se do této zavedené struktury komponenty zasahovat co nejméně a navíc by tyto informace mohly posloužit správci aplikace. Ten by však musel mít přístup přímo k DB, kde by mohl kontrolovat aktivitu uživatelů, dlouho neaktivní například mazat atd. Ovšem s touto variantou není v systému počítáno, takže pro něj není ani žádná podpora či uživatelské rozhraní.



Obrázek 13 – vztahy entity Users

UsersInRoles

Poslední z již zmiňovaných databázových tabulek spojených s komponentou pro přihlašování a správu uživatelů je entita UsersInRoles. Jedná se o jakousi asociační tabulka pro spojení tabulek Users a Roles, jejíž schéma je znázorněno na obr. č. 14.

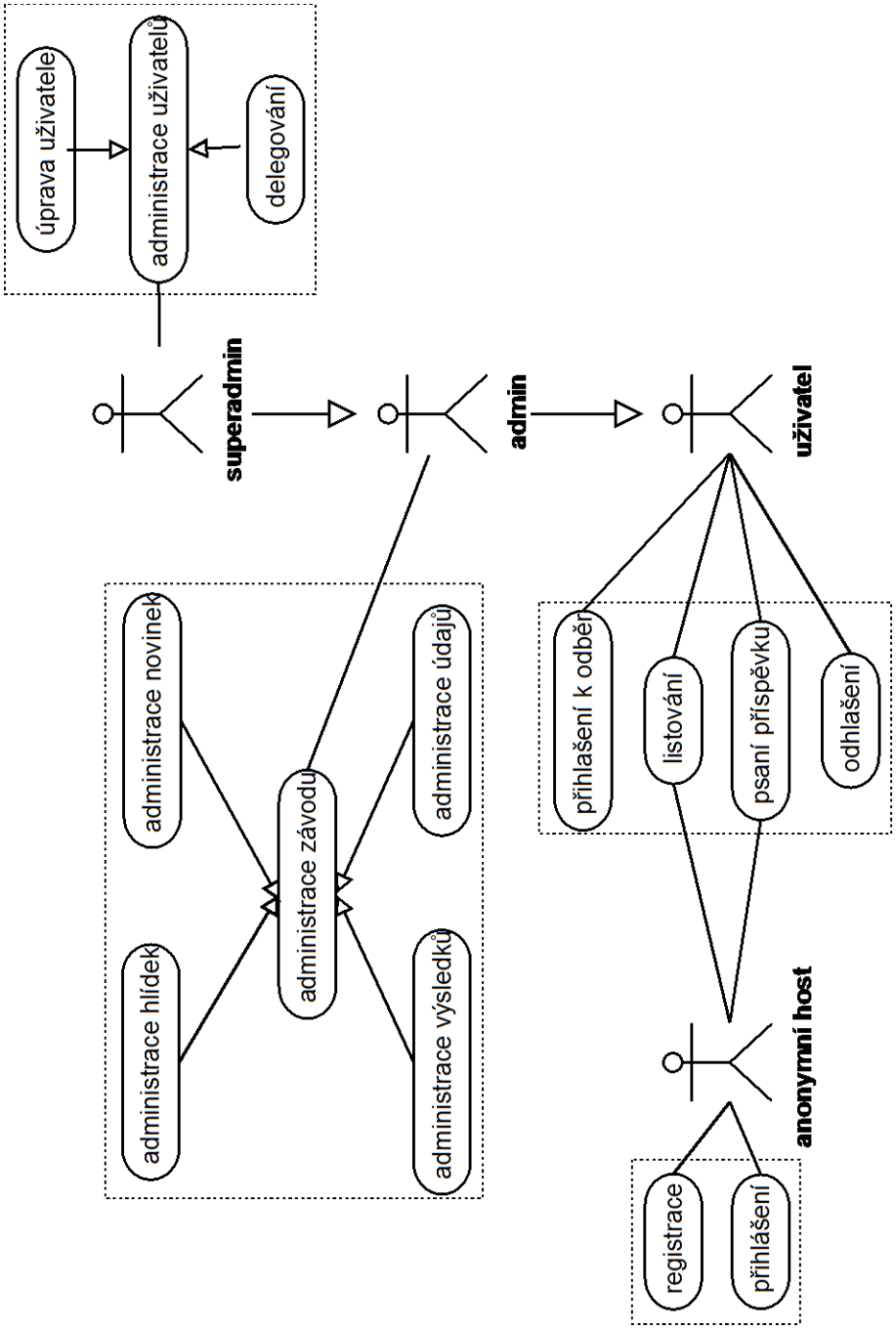


Obrázek 14 – provázání entit UsersInRoles a Users, Roles

3.3 Role uživatelů v systému

Už bylo zmíněno, že systém podporuje práci uživatelů zařazených do takzvaných rolí. Tyto role opravňují uživatele k určitým úkonům a naopak některé operace jim zakazují. Již z podstaty systému vyplývá, že bylo třeba navrhnout je tak, aby byly co nejtransparentnější a uživatelsky přístupné. Byly zvoleny celkem tři role *superadmin, admin, uživatel*.

Pro modelování jsem využil use-case diagram. Toto grafické zobrazení systému (viz obr. 15) nepopisuje samotný technický pohled, jako spíš rozsah jednotlivých rolí, do nichž jsou uživatelé systému zařazeni a činnosti, které tyto skupiny mohou vykonávat.



Obrázek 15 - celkový pohled na Use-Case diagram

3.3.1 Superadmin

Typ *superadmin* reprezentuje uživatelský účet s nejvyšším stupněm oprávnění. Měl by jím být člověk, jenž dbá nad celkovým stavem systému a řídí ostatní uživatele. K jeho právům patří i přidělování rolí ostatním uživatelům. Z podstaty návrhu je zřejmé, že superadmin má přehled o všem co se v systému děje, není tudíž potřeba, aby sám měl přidělenou nějakou lokalitu, kterou bude spravovat. Tato role je rozšířením role *admin*. Uživatel příslušející do zmiňované kategorie deleguje jiné uživatele do rolí systému, přiděluje jim oprávnění, lokalitu a má přístup ke všem položkám aplikace. Je jasné, že uživatelů, jež dosáhli tohoto zařazení, by nemělo být mnoho z důvodu udržitelnosti spojitosti mezi nimi a jednoduché komunikace týkající se například problému.

3.3.2 Admin

Rolí *admina* je pověřen člověk zodpovědný za prezentaci každého jednotlivého závodu a zodpovídá za něj. Na tento závod jej deleguje *superadmin*. Ať již je to dočasným přeřazením z role *uživatel* nebo vytvoření nového účtu a přímým zařazením do *admin* role. Není stanovena žádná komise starající se každoročně o jednu určitou geografickou oblast. Tato pořadatelská činnost se přesouvá napříč organizačními jednotkami Junáka a není pevně stanovena. Z tohoto faktu lze vyčíst, že uživatel se sice může stát adminem, ale tato oprávnění by mu po čase měl superadmin opět odebrat. Závod se tím pro něj stane uzavřeným, pro superadmina však zůstává stále přístupným.

3.3.3 Uživatel

Nejnižším stupněm, jehož může návštěvník webových stránek dosáhnout, je role *uživatele*. Dalo by se říci, že prakticky jedinou výhodou oproti neregistrovanému návštěvníkovi, kterou takto přihlášený uživatel získá, je možnost zaslání upozornění, že byly do systému vloženy výsledky závodu. Podle toho, jak je celá aplikace navržena a koncipována se tato varianta jevila jako plně dostačující. Další výhody přibudou s případným rozšířením aplikace o nové funkce a vlastnosti.

3.3.4 Anonymní host

Samozřejmě existuje možnost, že ne každý uživatel se chce registrovat, nebo nepotřebuje zaslat informační emailu. V tom případě mu nesmí být zamítnut přístup k celé aplikaci. Nepřihlášený uživatel může, stejně jako přihlášený, procházet všemi informačními úrovněmi aplikace.

3.4 Přihlašování uživatele

Platforma .NET a prostředí ASP.NET nabízí možnost správy uživatelů. Tato funkcionalita je již vestavěná a není tedy nutné dodatečně instalovat různé další knihovny apod. Nicméně po přečtení několika článků jsem dospěl k názoru, že pro potřeby mojí aplikace je tento vestavěný modul příliš složitý a těžkopádný. Sáhl jsem proto po alternativním řešení, jímž je *Altairis Simple ASP.NET SQL Provider* [7]. Tento *membership* a *role provider* pracuje na stejném principu jako originál, nicméně je nepoměrně zjednodušen a připraven pro co nejjednodušší implementaci do již existujícího systému. Membership provider zajišťuje autentizaci uživatele, stará se tedy o ověření, zda je daný uživatel již registrován a ověřuje jeho přihlašovací údaje. Naproti tomu Role provider se stará o autorizaci uživatele – tedy o oprávnění k určitým úkonům a zdrojům obecně. Jak už samotný název poskytovatele napovídá, informace o uživatelích a jejich atributech jsou ukládány do relační databáze, odkud jsou při přihlášení zase naopak přebírány pro pozdější využití. V této oblasti nabízí ASP.NET, na rozdíl např. od PHP, velmi dobrou vestavěnou podporu.

3.5 Předávání parametrů

Pro práci v systému je třeba nějakým způsobem zajistit předávání potřebných parametrů. Ať již při přechodu mezi jednotlivými stránkami, nebo mezi jednotlivými odesláními údajů na server a znovunačtením téhož webového formuláře. Ve své implementaci využívám tyto hodnoty například k identifikaci aktuálně prohlíženého či upravovaného závodu. Při editaci uživatele si tak udržuji identifikátor, pomocí něhož lze provést aktualizaci dat apod.

3.5.1 Query string

První možností jak předávat název proměnné a její hodnotu je přidání těchto informací do *query stringu* (jeví se tak jako součást URL). Obecně tuto možnost nepreferuji, protože uživatel na první pohled vidí některé věci, které by mu dle mého názoru, měly zůstat skryté. V aplikaci jsem však udělal dvě výjimky.

Prvním případem, kde je využito tohoto způsobu, je kontrola, zda má uživatel zapnuté cookies – malé textové soubory uložené na straně klienta, pomocí nichž si můžeme uchovat některá data. Provádí se vždy při načtení startovní stránky. Nepřišel jsem totiž na jiný způsob, jímž by šlo daný problém vyřešit. Samozřejmě, ASP.NET podporuje zjištění údajů podle prohlížeče, ale to nám prozradí pouze, jestli daný prohlížeč cookies podporuje obecně, ne jestli je má uživatel skutečně povoleny. Proto zkusím odeslat zkušební cookie a následně ji také vyzvednout. Tato kontrola je

pouze pro potřeby uživatele, aby mu, v případě vypnutých cookies, dala tuto skutečnost najevo v podobě varovné hlášky. Přihlášení do systému totiž v tom případě nelze provést.

Druhým z nich je kniha návštěv, kde jako součást query stringu předávám aktivní stranu výpisu příspěvků, na které se uživatel nachází. V tomto případě si myslím, že to bylo nejmenší zlo a uživateli vlastně neříkám nic, co by sám nevěděl.

Tato metoda má ale také svoje klady, neboť je velmi jednoduchá na implementaci i pochopení a prakticky nezatěžuje webový server.

3.5.2 Sessions

Kromě přímého poslání hodnoty v adrese odkazu na cílovou stránku, jež není v této aplikaci téměř využito, je v systému zastoupena i technika *sessions*, která je naopak využita velmi často. Její funkcionalita je založena na unikátním ID sezení, které je udržováno na straně klienta – konkrétně v souboru cookie. Existuje několik variant, kde udržovat vlastní obsah (v paměti na straně serveru, v databázi,…). V tomto konkrétním případě je udržován v paměti na serveru, tudíž k němu uživatel nemá přímý přístup a nemůže jej, ať již záměrně nebo nevědomky, editovat či smazat. Bohužel, jako všechna, má i toto řešení své nevýhody. Zřejmě tou hlavní je nutnost mít v prohlížeči zapnuté cookies. U většiny uživatelů tomu tak je, nelze na to však spoléhat. Toto nastavení je kontrolováno při vstupu do aplikace. Druhým nedostatkem je další zatížení serveru. Ovšem v případě této práce zmíněný fakt opomenu, protože se nepředpokládá nijak extrémně vysoká návštěvnost serveru, kdy by musel server odpovídat stovkami operací během krátké doby a přitom zaznamenávat stav každé session.

Pokud bychom se chtěli vyhnout ukládání unikátního ID sezení do souboru cookie, jde to obejít, ale potom je toto ID šířeno jako součást URL adresy. Varianta bez užití cookie však, dle mého, dost znepřehledňuje adresy odkazu, a proto není v této práci využita.

Při použití session je třeba dát pozor na tzv. *session-stealing* – techniku útoku na session přihlášeného uživatele. Útočník se snaží získat její identifikátor a zneužít jej při vydávání se za oprávněného uživatele. Proto je ukládána i IP adresa a průběžně s každým požadavkem na server je tento údaj kontrolován. Pokud se prokáže, že příkaz pochází z počítače, jehož IP adresa je jiná než v session, je odmítnut z důvodu možného útoku.

3.6 Bezpečnost

Sice tato práce není nijak spojena například s převodem financí nebo nějak zvlášť citlivých osobních dat, ale nelze se problematice bezpečnosti vyhnout. Navíc je třeba hlídat díry vedoucí ke snadnému vniknutí do IT prostředí na kterém bude aplikace hostována, aby se skrz naši aplikaci nebylo možné jednoduše dostat na hostingovou službu, databázový server apod.

ASP.NET v této oblasti nabízí celou řadu možností. Od základních prvků pro zabezpečení aplikace až po obsáhlé komponenty některých třetích stran, které využívají sofistikovaných technik.

Já jsem se zaměřil na dvě oblasti přímo související s aplikací, jež by se mohly stát terčem útoku.

Jedná se o:

- administrační sekce, která umožňuje editaci závodů, výsledku,…
- databáze, uchovávající data

3.6.1 Zabezpečení administrační sekce

Celá administrační sekce se nachází ve vlastním adresáři. Jednak je to pro zpřehlednění struktury celého webu a druhým důvodem je bezpečnost. ASP.NET totiž dokáže ohlídat jednotlivé složky a přístup k nim. Celá aplikace obsahuje speciální konfigurační soubor *web.config*. Ovšem v každé složce umístěné na webovém serveru lze mít jakýsi *lokální* konfigurační soubor s omezeními týkajícími se pouze konkrétní složky. V tomto souboru s nastavením si totiž lze zvolit, že jediným uživatelem, který dostane přístup do složky, bude např. uživatel „*t35t0v@c1_uz1v@t3l*“. Pokud chceme umožnit přístup celé skupině uživatelů, nechceme určitě všechny vypisovat jednotlivě. Navíc při přidání nové osoby soubor vždy aktualizovat. Proto existuje podpora pro mechanizmus zmíněný již v kapitole věnované přihlášení uživatele. Je to role management. Ten totiž ve spolupráci s konfigurační službou dokáže povolovat přístup pouze určité skupině lidí příslušející do jedné ze spravovaných rolí. Složka je chráněna a nastavena tak, aby se do ní dostal pouze přihlášený uživatel, který navíc musí splňovat podmínku přítomnosti v roli admin či superadmin. Velkou výhodou je také filosofie této techniky, kdy je zároveň chráněn i obsah složky, tzn. není nutno v hlavičce stránky explicitně potvrzovat restriktivní opatření proti nežádoucím uživatelům, vše je řízeno na úrovni adresáře.

3.6.2 Zabezpečení databáze

Druhou rizikovou oblastí je databáze. Útok na ni by mohl zhroutit celou aplikaci. I když v tomto případě není příliš pravděpodobné, že by útočník chtěl získat nějaká data nebo zbortit web, je vždy lepší počítat s horší variantou a tématem se alespoň okrajově zabývat. Velkou slabinou aplikace bývají interakce s uživateli. Pokud bychom nekontrolovali vstup, který nám odesílá klient, mohlo by dojít k neoprávněné úpravě dat nebo dokonce vymazání databáze.

Nejčastěji užitou technikou takového útoku bývá tzv. SQL injection. Je to vlastně podstrčená část dotazu na databázi, ať už výběrového nebo jakéhokoliv jiného typu. Tato technika je použitelná jen v tom případě, že tvůrce webu špatně zabezpečil aplikaci a špatným způsobem přebírá vstupy od uživatele. Tedy pokud použije klasickou konkatenaci částí dotazu a na místo konkrétních údajů rovnou umístí data zaslaná útočníkem.

Naštěstí ASP.NET řeší tuto problematiku velice dobře a využívá parametrizované dotazy na databázi, kdy si sám objekt *SqlCommand* pohlídá, aby předávaný parametr neobsahoval rizikové sekvence. Výsledkem takto zadávaných dotazům by již měla být bezpečná práce s databází. Použití upraveného dotazu s parametry není složité:

```
SqlCommand sqlcomm = new SqlCommand();
sqlcomm.CommandText =  "SELECT id FROM logins WHERE username =
    @username AND password = @password";
sqlcomm.Parameters.AddWithValue("@username", username);
sqlcomm.Parameters.AddWithValue("@password", password);
```

Z důvodů zmíněných výše, jsem se snažil používat při práci hlavně tyto parametrizované dotazy, obzvlášť pokud se jednalo o vstupy, jež vyplňuje uživatel sám. Dalšími bezpečnostními nastaveními, například na straně SQL serveru, jsem se nezabýval, neboť tato nastavení jsou zcela v režii provozovatele hostingové služby.

3.7 Vstup a výstup generovaných dat

Jak již bylo naznačeno v úvodu této práce, není jejím výsledkem pouhý webový informační systém pracující na straně serveru. Měla být zajištěna i jeho spolupráce s offline verzí, podporující zadávání dat přímo v místě závodu. To vše s možností co nejjednodušší formou tento nově vytvořený obsah prezentovat na webu a udržovat jej v nezměněné podobě.

3.7.1 Výstup šablon

Výstupem šablon je myšleno generování určité kostry užité při vyplňování v tlustém klientovi. Tato šablona je generována přímo SQL serverem, protože již sám o sobě má pro generování dat do XML formy velmi dobrou podporu. Není tedy užito žádného externího konvertoru nebo jiné komponenty. Jedná se o XML soubor, se kterým tlustý klient dokáže dále pracovat a číst i vkládat potřebné údaje. Výsledná šablona má název

zavod-<*zavodID*>.xml

tedy např. zavod-2.xml

ukázka vygenerované šablony[2]:

```
<zavod ID="2" typ="4" start="2008-04-25T00:00:00" konec="2008-04-
     26T00:00:00" misto="Dolní Dobrouč, skautská klubovna">
    <hlidky hlidkaID="1>
        <mesto>Dolní Dobrouč</mesto>
        <kluci>1</kluci>
        <nazev>Delfíni</nazev>
    </hlidky>
    <hlidky hlidkaID="2“>
        <mesto>Letohrad</mesto>
        <kluci>1</kluci>
        <nazev>Kamzíci</nazev>
    </hlidky>
</zavod>
```

Většina názvů atributu je dostatečně popisná. Mohlo by se zdát, že jsou generovány i nepotřebné informace, např. start a konec závodu, je to však z toho důvodu, že tyto informace jsou využity při konečném exportu výsledků k zobrazení hlavičky. Element *kluci* určuje kategorii, do které hlídka patří, zda skauti, nebo skautky a *typ* určuje typ závodu – oblastní kolo, krajské kolo nebo celostátní. Z příkladu je však vidět, že udržujeme opravdu jen základní informace, které dále rozšiřuje tlustý klient.

3.7.2 Vstup dat

Vstupem dat v této kapitole se myslí uživatelský vstup v podobě importu výsledků závodu, tedy zpracování XML souboru. V něm by mohlo dojít ke snadné záměně a přiřadit tak výsledky k jinému závodu. Proto je třeba data před jejich zveřejněním a zpřístupněním uživatelům ověřit. K tomu je využita kombinace několika technik. Všechny tyto kontroly se provádějí na straně webového serveru. Soubor, který se uživatel pokusí přidat ke konkrétnímu závodu, je uložen v dočasném adresáři.

Prvním stupněm je ověření pravosti souboru. To znamená, že je kontrolováno, zda soubor je opravdu totožný s tím, který vyprodukoval tlustý klient a který by měl uživatel v nezměněné podobě předat klientu tenkému. Formát názvu souboru, generovaného tlustým klientem je následující:

`vysledkyZavodu-<`*`hash`*`>.xml`

tedy např: `vysledkyZavodu--1884286271.xml`

[2] Kompletní XSD schéma šablon je součástí příloh této práce

Proces ověření je realizován pomocí porovnání řetězce znaků (tzv. *hash řetězce obsahu souboru*), který je součástí názvu souboru s hash řetězcem vytvořeným ze skutečného obsahu doručeného na server. Hodnota hash je počítána jednosměrnou hash funkcí, která určitým způsobem vygeneruje zdánlivě nesouvislý řetězec znaků. Jeho hodnota však neříká nic o samotném obsahu souboru. Proto je na serveru opět spočítán z doručených dat a porovnán s původním. Pokud jsou oba řetězce shodné, máme správný obsah a pokračujeme s dalšími kontrolami.

Následujícím stupněm zjištění korektnosti je porovnání ID závodu obsaženého v souboru s ID závodu, který se uživatel rozhodl editovat (přidat mu výsledky). V XML souboru je tento údaj obsahem atributu *ID* v kořenovém elementu dokumentu. Webový server se podívá do session, kde je uložen i identifikátor aktuálně upravovaného závodu. Ověřením shodnosti těchto čísel lze odvodit, zda se uživatel nepokouší nahrát data k jinému závodu.

Pokud uživatel poskytl korektně vygenerovaný soubor a vybral správný závod, otestujeme ještě poslední kontrolní bod. Tím je správnost hodnot a validace celého dokumentu. Existuje několik možností jak takového stavu dosáhnout. Já jsem zvolil validaci proti XSD – *XML Schema Definition* [4]. Jde o schéma XML dokumentu, které nám říká, jak by měl kontrolovaný dokument vypadat. Definuje povolené elementy a atributy, jejich počet, pořadí a datový typ obsahu. Navíc můžeme určit, zda je daný prvek povinným nebo jen volitelným – tedy zda je dokument platný i bez uvedení této hodnoty, atd. Další výhodou oproti staršímu způsobu validování pomocí DTD je skutečnost, že XSD je samo zapsáno pomocí jazyka XML. Není tedy nutné se učit syntaxi dalšího jazyka.

Projde-li soubor s výsledky všemi kontrolami, lze o něm prohlásit, že je platný a postoupit jej ke zveřejnění.

3.7.3 Výstup dat

Po úspěšném otestování je celý nahrávaný soubor překopírován z dočasného adresáře do složky s výsledky závodů. V této složce jsou uloženy všechny již přidané výsledky pro jejich případné snadné dohledání. Samozřejmě pohled na strohý XML soubor není ideální, proto je třeba jeho obsah rozumnou formou sdělit uživateli.

Ideálním řešením pro web je prostý HTML dokument. Nabízelo se několik možností jak tohoto stavu dosáhnout. Například generovat výslednou stránku s každým požadavkem pocházejícím od jednotlivých uživatelů. Tato varianta se mi zdála zbytečná z důvodu naprosté stálosti informací. Pokud by se obsah zdroje dat – v našem případě XML souboru – často měnil, pak by dynamické generování bylo zcela jistě na místě. Z důvodu toho, jak je celá aplikace koncipována a nastavena, není důvod se domnívat, že se zdroj dat bude vůbec kdy měnit. Proto jsem se rozhodl vygenerovat jednu statickou stránku a tu přidat do stejné složky s výsledky, jako již zmiňovaný datový zdroj. Toto řešení není určitě ideální, ovšem požadavky na něj kladené plní dobře a je plně dostačující. K vlastní transformaci z hrubého XML formátu do výsledné HTML podoby je použito XSLT procesoru

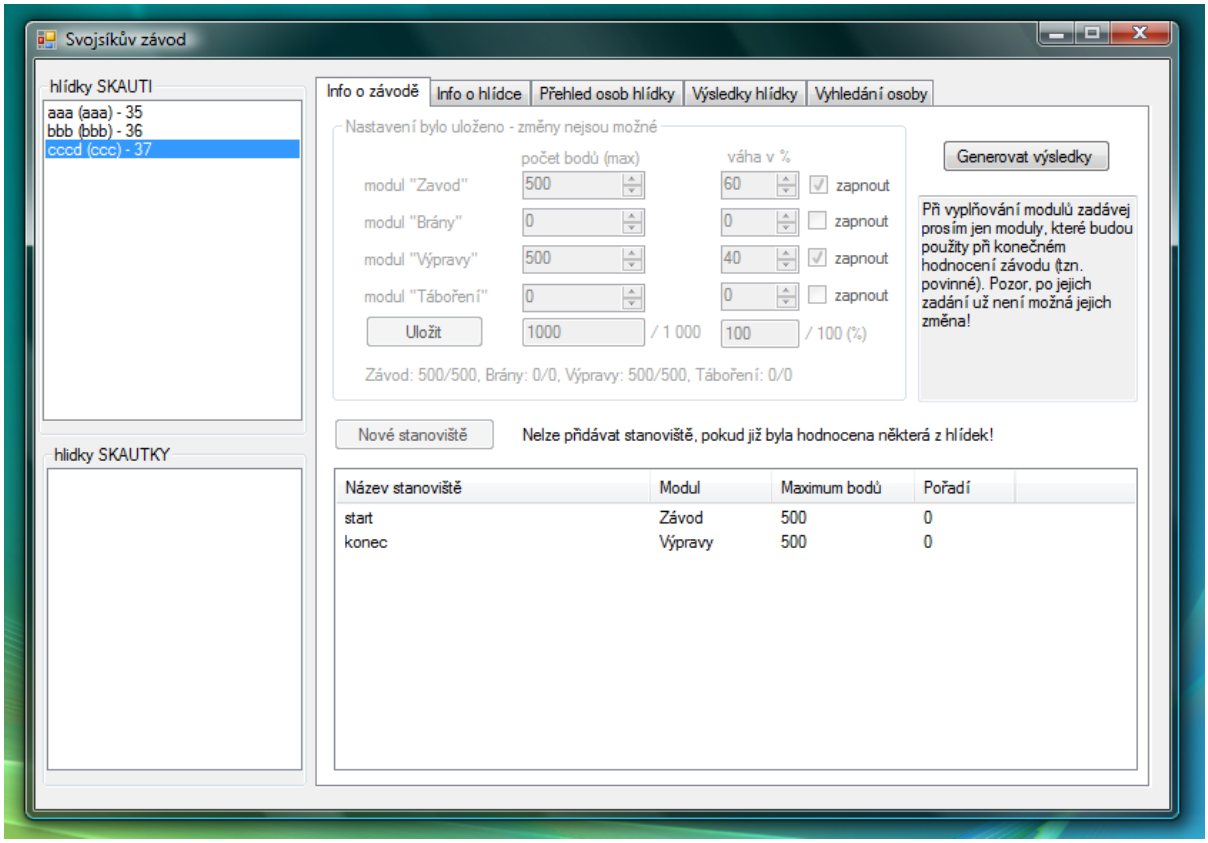
(*eXtensible Stylesheet Language Transformations*) [9]. Jedná se o soubor pravidel, podle kterých je XML soubor převeden do požadovaného tvaru. Tímto tvarem nemusí být zdaleka pouze webová stránka, k čemuž jsem jej použil já, ale prakticky jakýkoliv jiný textový formát či datová struktura. Často je využíván například k transformaci na jiný XML dokument nebo SQL dotazy. V počátečním návrhu jsem se zabýval i myšlenkou uchovávat data v databázi, ale nakonec byl použit XML formát. Ať již z důvodu zmíněné stálosti dat nebo nepřítomnosti jakékoliv funkce, která by mohla nějak využít potenciál, jenž oproti uložení v XML databáze nabízí. Pokud by se ale v budoucnu do systému vkládala nová funkcionalita – např. prohledávání historie výsledků závodů a bylo by nutné převést výsledky z XML do databáze, stačilo by využít jeden XSLT styl pro vytvoření příslušných vkládacích dotazů.

Nedostatkem, který se mi stále nepodařilo vyřešit, však zůstává fakt, že jsem nepřišel na způsob, jakým do stránky s výsledkovou tabulkou dostat příslušný DOCTYPE. Když jsem zkoušel provést XSLT transformaci pomocí simulačního nástroje, vše proběhlo tak jak mělo – tedy i se správným vložením DOCTYPE do zdrojového kódu. Pokud jsem ale ten samý transformační předpis použil pro převod na serveru, DOCTYPE zmizel. Stránka je i tak plně funkční a zobrazitelná ve všech třech hlavních webových prohlížečích pod OS Windows. Bohužel však při kontrole validity nahlásí chyby.

4 Tlustý klient

Úkolem tlustého klienta je usnadnit pořadateli zpracování výsledků SZ již během dobíhání jednotlivých hlídek do cíle i bez nutnosti připojení k internetu a tenkému klientovi. Největší výhodou by měla být snadná možnost publikovat výsledky do již připraveného systému, přístupného všem zájemcům. I tato část práce je realizována pod .NET frameworkem, programovacím jazykem byl zvolen C# [3].

Opět jsem se snažil o co nejlepší uživatelskou přístupnost, aby důležité informace byly na očích a časté operace se daly realizovat co nejrychleji. Tomu by měl napomoci i záložkový ovládací prvek, který na minimálním prostoru poskytuje větší užitnou plochu. S obsluhou by neměl být žádný problém, protože uživatelské rozhraní je v klasickém stylu aplikací operačního systému windows (viz obr. 16).



Obrázek 16 - pohled na úvodní obrazovku tlustého klienta

4.1 Ukládání dat

V počáteční fázi návrhu bylo v plánu, využít jako úložiště dat pro tlustého klienta databázi. Výběr se zúžil na vestavěné databáze, protože by bylo neúnosné a zbytečné kvůli tak jednoduché aplikaci

instalovat celý SQL server, ať již z důvodu složitosti, tak i plýtvání zdrojů, které by stejně nebyly zdaleka plně využity. Po zhodnocení všech pro a proti jsem se ale nakonec rozhodl vypustit databáze úplně a pro ukládání veškerých údajů využít XML soubor. Hlavním motivem mi bylo celkové zjednodušení aplikace. Přenos dat mezi tenkým a tlustým klientem je taktéž realizován pomocí XML souboru, takže odpadá několik mezikroků v převodu (databáze ↔ XML). Na druhou stranu se zvyšuje riziko, že někdo neoprávněně zasáhne do výsledného souboru, což by bylo při použití databáze rozhodně složitější.

Vlastní průchod datovým souborem je realizován za pomoci jazyka XPath - *XML Path Language*. Tímto nástrojem lze adresovat jednotlivé elementy dokumentu nebo celé jeho části. Dále umožňuje pracovat s jejich obsahem a velkou výhodou je jeho dobrá dokumentace a velké množství návodů [8, 9] plynoucí z podpory konsorciem W3C.

Při požadavku na načtení dat je dokument vždy kontrolován a potřebný obsah načítán do příslušných textových polí, dalších ovládacích prvků nebo jiným způsobem zpracován. Pro navigaci a výběr elementů či atributů je použito výše zmíněného jazyka XPath. Tato varianta zajišťuje stálou aktuálnost i validitu zdrojového souboru. Při opačném procesu – tedy ukládání (ať změn již existujících nebo vkládání nových záznamů) jsou naopak data okamžitě po události indikující požadavek na zápis fyzicky zaznamenána na disk. Tím minimalizujeme možné riziko ztráty dat z důvodu např. výpadku proudu, výskytu neočekávané chyby apod.

4.2 Bezpečnost

Bezpečnost v tlustém klientu není řešena tak detailně jako u klienta tenkého, protože k němu bude mít přístup nepoměrně menší skupina lidí.

4.2.1 Přihlášení do aplikace

Ačkoliv se jedná o přístup k celému systému pro zadávání výsledků, přihlašování uživatele jsem v tlustém klientu prakticky neřešil. Důvodů je několik. Tím prvním je zabezpečení samotného operačního systému, který při svém spuštění po uživateli žádá heslo. Takže nepovolaná osoba by prakticky neměla mít přístup ani do samotného počítače k aplikaci. Druhým důvodem je kontrola při předávání výsledku do webové části informačního systému, kdy je po uživateli opět požadován přístupový kód, bez kterého se nelze dostat do administrační části. Třetí důvod je ryze praktický, a sice že počítače, které budou k tomuto účelu využívány, jsou majetkem uživatele a je v jeho vlastním zájmu hlídat si své osobní věci. Samozřejmě bylo možné opatřit i tuto část sofistikovanější ochranou před neoprávněným útokem, to ale nebylo zadáním práce. Pokud by bylo vyžadováno, lze jej doimplementovat v pozdější fázi.

4.2.2 Vstup dat

Aplikace sama o sobě potřebuje ke svému běhu dodat XML šablonu, do níž bude doplňovat vkládaná data od uživatele. Pokud se programu nepodaří ji najít, pouze to oznámí uživateli a ukončí se. Není tedy jeho úkolem tuto šablonu vytvářet, pouze ji plnit daty, šablonu generuje tenký klient. Jeho činnost a popis výstupu je popsán v jiné kapitole.

Při načítání šablony je nutné zkontrolovat strukturu předávaných dat, protože během vkládání nových hodnot je striktně vyžadován konzistentní tvar šablony. Toto ověření je opět prováděno technikou validace proti XSD schématu XML souboru. Popisovat XSD zde nebudu, popis a princip činnosti je popsáno v kapitole věnující se tenkému klientovi. Z toho je také zřejmá výhoda plynoucí z implementace obou částí pod .NET frameworkem. Způsob zpracování je totiž velmi podobný pro oba klienty.

4.3 Generování výstupu

Pokud měl být splněn požadavek na snadnou přenositelnost, bylo třeba upravit výstup do srozumitelné podoby. Z tohoto důvodu je výstupem aplikace trojice souborů:

- HTML stránka
- XLS sešit
- XML soubor

Někoho možná napadne, proč je opět, stejně jako v případě tenkého klienta, generována webová stránka. Důvody jsou hned dva. Prvním je možnost, že uživatel, na jehož počítači aplikace poběží, nebude mít nainstalován balík Microsoft Office. Pokud by tedy byl výstup směřován pouze tímto směrem, přišel by o možnost vyhodnocení závodu. Druhým důvodem je možnost okamžitého umístění stránky na weby jednotlivých skautských středisek.

Dalším výstupním tvarem, který již byl zmíněn, je sešit programu Microsoft Excel. Na první pohled by se mohlo zdát, že převod do takového formátu bude komplikovaným, nicméně Excel je natolik silným programem, že si poradí i s otevřením souboru, který není nativně vytvořen jím samotným. Jedním z typů, se kterým si rozumí je i obyčejné HTML. Mírným upravením stylu pro převod XML vstupu lze zajistit přijatelné zobrazení v Excelu. Z těchto změn mám konkrétně na mysli přidání pravidel kaskádových stylů, které jsou při exportu do HTML uloženy v externím CSS souboru, ignorování zápisu verze XML a několik drobných změn ve struktuře celé stránky. To z důvodu, že Excel pracuje se stránkou jako by to byla jedna tabulka, popřípadě s dalšími tabulkami vnořenými. Nepříjemností, která byla nakonec naprosto triviální, jen jsem ji nedokázal delší dobu odhalit, byla nepřesnost Excelu při interpretaci obsahu buňky tabulky. V některých případech, pokud v ní došlo k zalomení řádku pomocí HTML elementu `<br />`, začala se její číselná hodnota

nepředvídatelně měnit. Navíc se nejednalo o odchylky v řádu desetin bodů ale i stovek, někdy se hodnota dostala do záporných hodnot apod. Za jakých předpokladů se tak dělo se mi nepodařilo vypozorovat. Jako řešení se nakonec ukázalo tyto elementy konce řádku zcela vynechat.

5 Možná rozšíření systému

Výsledný systém má spoustu rezerv, které by se daly odstranit jejich postupnou obměnou. Některé funkce by mohly být přidány a zvýšit tak celkovou užitnost aplikace.

Celkově by bylo vhodné zefektivnit práci s daty. Databáze zdaleka nevyužívá svůj potenciál, stejně jako prostředky .NETu pro práci s XML soubory. Při použití databáze by bylo lepší využívat více uložených procedur. Tak by se i část zatížení webserveru přenesla na server databázový. Pro složitější dotazy, jež načítají např. data do administračního rozhraní, bych nyní využil pohledů atd. Při práci s XML bych se snažil zefektivnit především počet přístupů (IO operací) na pevný disk. V současném systému je při každé změně bodu třeba tyto změny uložit.

Pokud bychom uvažovali vylepšení pro tlustého klienta, dal by se doplnit například modul, jenž by umožňoval efektivnější správu účastníků závodů a staral by se o umisťování osob do připravených ubytovacích kapacit. Měli bychom pak přehled o jednotlivcích i celých hlídkách. Dal by se implementovat i komplexnější vyhledávač osob nebo tisk diplomů apod. V současné verzi je nutnost umístit šablonu generovanou tenkým klientem přímo do složky tlustého klienta a přejmenovat ji na předem známý tvar. Podobně výstup je exportován do složky programu. Tato varianta byla zvolena z důvodu jednoduchosti a její omezení by šla poměrně jednoduše eliminovat přidáním dialogu pro výběr konkrétního souboru.

V případě rozšíření tenkého klienta by bylo vhodné doplnit redakční systém pro vkládání reportáží o celkovém průběhu závodu. Dalším prvkem, zvyšující informační hodnotu webu by byl prostor věnovaný každé z přihlášených hlídek, kde by mohla uvést prezentaci svého oddílu nebo střediska, kontakt, nabídku ubytování ve své klubovně apod. Tímto by vznikla i databáze samotných kontaktů pro snadné vyhledávání a řazením podle oblastí. Přepracovat by se dala i kniha návštěv, jež v současné verzi splňuje opravdu jen základní funkcionalitu. Přidána by byla podpora vkládání drobných ikonek, reakcí na příspěvky, zabezpečení proti spam robotům sofistikovanějšími prostředky typu *CAPTCHA* apod. Užitečnou funkcí, kterou bych v budoucnu doplnil, by bylo automatické označování postupujících účastníků se zkopírováním informací o hlídce do nadřazeného závodu.

Závěr

Hlavním cílem bylo zjednodušit práci při vyhodnocování výsledků Svojsíkova závodu a usnadnit jejich publikaci na webu. Dosud nebyly výsledky na jednom centrálním místě a situace se řešila buď založením nové internetové prezentace pro konkrétní ročník závodu, nebo publikací na stránce konkrétního pořádajícího střediska.

Celková struktura a nejasnosti, které vznikly během návrhu a implementace systému byly konzultovány se zástupci organizace Junák – svaz skautů a skautek ČR a výsledná aplikace by tak měla splňovat všechny požadavky na ni kladené.

Vlastní implementace byla rozdělena do dvou modulů – tlustého a tenkého klienta, kdy obě byly vytvářeny v prostředí .NET framework. Jako vývojový nástroj bylo použito MS Visual Studio 2008, programovacím jazykem byl C#. Dalšími prostředky pro úspěšnou realizaci byl jazky XPath, transformační styly XSLT, značkovací jazyk XML, validační schémata XSD, samozřejmě prosté (X)HTML a dotazovací jazyk pro databáze SQL. K výsledné podobě tenkého klienta bylo navíc využito kaskádových stylů.

Pro mě osobně bylo největším přínosem seznámení se s, pro mě, novou technologií, kterou byl ASP.NET a realizace projektu, který propojoval několik oblastí. Ať již to byla spolupráce webové a desktopové části, tak i použití XPath dalších výše vypsaných nástrojů. Velmi cenným pro mě je i získaný přehled nad použitými technologiemi.

Literatura

[1] Lacko, L. *ASP.NET a ADO.NET 2.0: hotová řešení*. 1. vydání. Computer Press, Brno, 2006. 385 s. ISBN 80-251-1028-1

[2] Prosise, J. *Programování v Microsoft.NET*. 1. vydání. Computer Press, Brno, 2003. 712 s. ISBN 80-7226-879-1

[3] Sells, Ch. *C# a Winforms: programování formulářů windows*. 1. vydání. Zoner Press, Brno, 2005. 648 s. ISBN 80-86815-25-0

[4] Skonnard, A., Gudgin, M. *XML - pohotová referenční příručka: referenční příručka programátora ke XML, XPath, XSLT, XML Schema, SOAP a dalším*. 1. vydání. Grada, Praha, 2006. 342 s. ISBN 80-247-0972-4

[5] *CodeProject* [online]. 1999-2008 [cit. 2008-05-06]. Dostupný z WWW: <http://www.codeproject.com/>.

[6] Microsoft Corporation. *MSDN Library* [online]. 2008. [cit. 2008-05-06]. Dostupný z WWW: <http://msdn.microsoft.com/library/>.

[7] Valášek, M., *ASPNET.CZ* [online]. 2000-2008 [cit. 2008-05-06]. Dostupný z WWW: <http://www.aspnet.cz/>.

[8] *W3Schools Online Web Tutorials* [online]. 1999-2008. [cit. 2008-05-11]. Dostupný z WWW: <http://www.w3schools.com>.

[9] *ZVON.org* [online]. 2000. [cit. 2008-05-10]. Dostupný z WWW: <http://www.zvon.org>.

Seznam příloh